

Пользовательская документация по
OmegaBot IDE

Санкт-Петербург

2021 г

Оглавление

Описание программы	3
Концепция и предназначение системы	3
Инструкции по установке	5
Описание интерфейса программы	7
Описание модулей программы	12
Управление	12
Порты	15
Проверки	16
Переменные/Константы	17
Математические операторы	20
Сенсоры	21
Серво	22
Светодиоды	23
Звук	23
Дисплей	24
Ориентация в пространстве	25
Моторы	25
Эндокеры	27
Коммуникации	27
Хранилище	28
Блоки кода	29

Описание программы

Данное решение позволяет составлять сложные алгоритмы и задавать команды Роботу без знания языка программирования, доступно для понимания как детей, так и взрослых. Программная среда Omegabot_IDE, специально разработанная для управления Роботом, позволяет писать алгоритм управления роботом OmegaBot путём перетаскивания в рабочую область цветных блоков с командами.

Концепция и предназначение системы

OmegaBot IDE – робототехническая платформа с программируемыми модулями. Платформа была разработана специально для сопровождения образовательных учреждений в обучении робототехнике и программированию. Благодаря интуитивно понятному интерфейсу и визуальному программированию у пользователя ОмгаБота будет возможность разобраться и быстро начать работать в данной среде. Среда работает под Arduino IDE, благодаря чему также можно будет пользоваться и процедурным программированием.

Программирование ОмгаБота может быть осуществлено как процедурным, так и визуальным программированием, то есть подходит как для новичков, так и для любителей платформы Arduino. Причем при визуальном программировании можно обучиться программированию процедурному, так как исполняемая программа генерируется в соответствии с рис. 1.

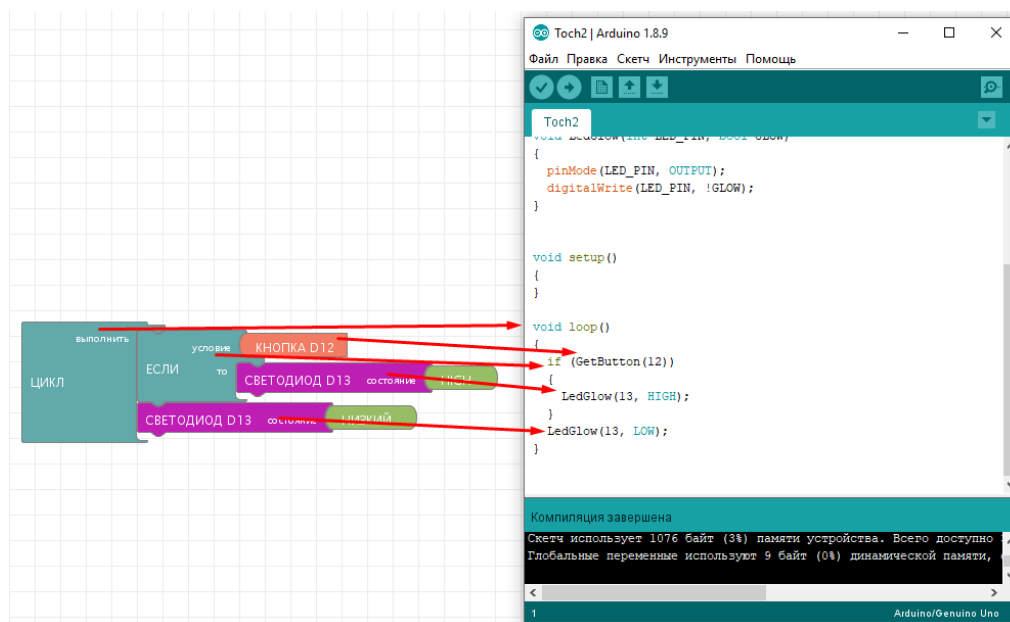


Рисунок 1 - пример работы OmegaBot IDE

Программа OmegaBot IDE - это дополнение Arduino IDE, в ней осуществляется визуальное программирование. После составления в ней программы, генерируется код в

Arduino IDE, соответствующий алгоритму, описанному блоками. Для каждого вида блока визуального программирования генерируется функция. Функция вызывается в программе каждым блоком, с параметрами, которые также задаются блоками.

Для начала работы необходимо:

1. Установить Arduino IDE
2. Установить OmegaBot IDE
3. Проверить работоспособность

Инструкции по установке

Необходимо скачать среду визуального программирования OmegaBot IDE. Скачайте инсталлятор с сайта <https://omegabot.ru/software> и установите на компьютер, как показано на рис. 2-3.

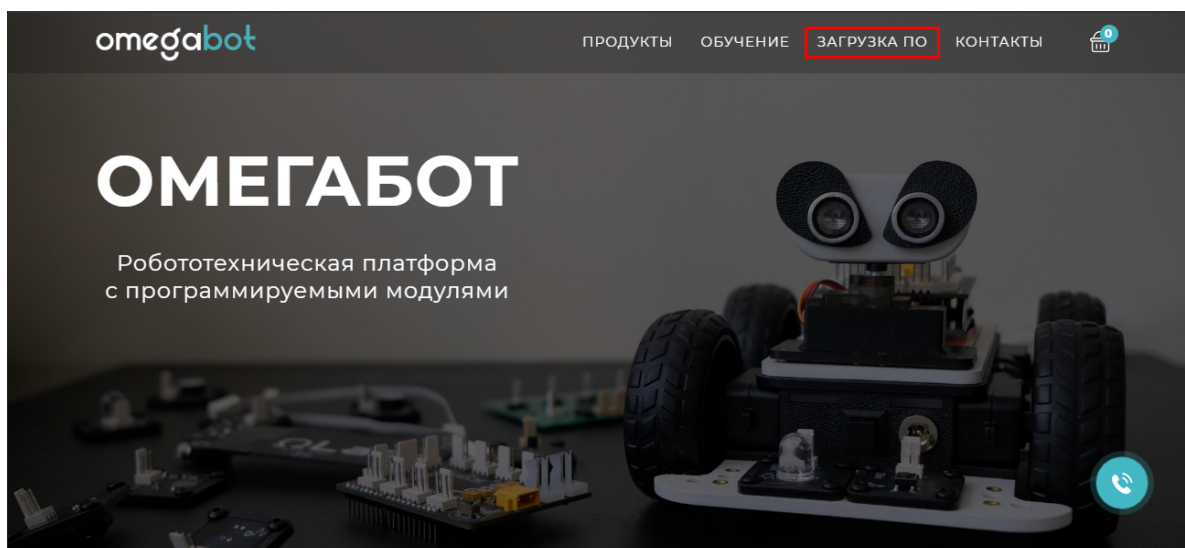


Рисунок 2 – Загрузка OmegaBot IDE

СКАЧАТЬ ОМЕГАБОТ IDE

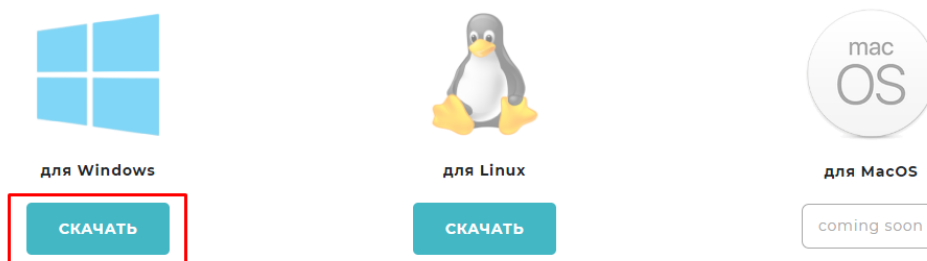


Рисунок 3 – Загрузка OmegaBot IDE

Далее следуйте в соответствии с шагами:

- 1) Откройте скачанный архив и распакуйте файлы в удобную вам директорию
- 2) Откройте установщик (файл OmegaBot_IDE-setup.exe)
- 3) Выберите язык установки и нажмите кнопку "ОК"
- 4) Arduino IDE:
 - Если у вас не был установлен Arduino IDE, то вам будет

предложено его установить, нажмите кнопку "Да"

- При нажатии кнопки "Нет", вы можете установить Arduino IDE

сами или заново открыть установщик OmegaBot IDE

и в этот раз выбрать кнопку "Да" для установки Arduino IDE

(В архиве с установщиком также находится папка с установщиком Arduino IDE)

- После установки Arduino IDE будет продолжена установка OmegaBot IDE

5) При выборе компонентов:

- Выберите вариант "Полная установка" и нажмите "Далее >"

- Или выберите все пункты и нажмите "Далее >"

6) Нажмите "Установить" и дождитесь окончания установки среды, после чего нажмите кнопку "Завершить"

7) Запустите на рабочем столе (или из меню "Пуск") Arduino IDE

8) Для перехода в среду ОмегаБота откройте в Arduino IDE вкладку "Инструменты" -> "OmegaBot_IDE"

9) В поле авторизации введите ваш ключ продукта (серийный номер робота: можно найти наклейку на дне коробки или на батарейном отсеке робота) и нажмите кнопку "ОК"

Описание интерфейса программы

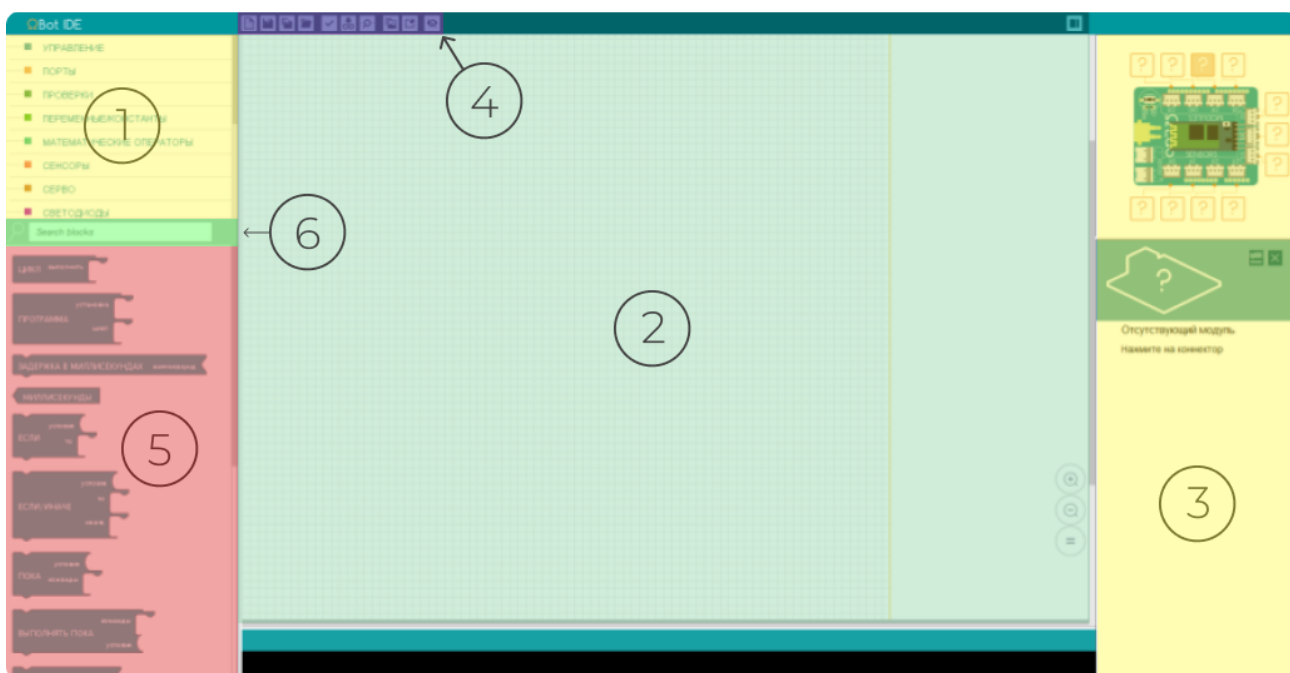


Рисунок 4 - Описание интерфейса

В соответствии с Рисунком 4 описание рабочих зон программы:

1. Панель выбора блоков

В первой рабочей зоне находятся разделы, содержащие в себе различные команды и/или элементы визуального программирования, находящиеся в пятой рабочей зоне. Т.е. выбрав пункт «Порты» будут доступны такие элементы, как: «УСТ. ЦИФРОВОЙ ПОРТ», «ПЕРЕКЛЮЧИТЬ ЦИФРОВОЙ ВЫХОД» и т.д.

2. Холст, где будет писаться программа.

Внимание: выполняются только те блоки, которые подключены в «Цикл» или «Программа».

Во вторую рабочую зону перетаскиваются блоки и выстраивается алгоритм работы. В некоторых блоках можно изменить некоторые переменные, например, количество секунд, пока светодиод горит или выбрать состояние диода. (Пример на рис. 5).



Рисунок 5 – Выбор состояния диода

Также при помощи следующих трёх элементов можно отдалить или приблизить алгоритм или поставить автоприближение (рис. 6).



Рисунок 6 – Масштабирование алгоритма визуального программирования

Также в рабочей зоне находится консоль состояния, через которую можно узнать о работе программы. Данная консоль показана на рис. 7.

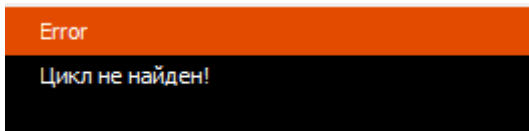


Рисунок 7 – Пример одной из ошибок

3. Подключения к контроллеру.

Здесь отражены подсказки при составлении программы.

В третьей рабочей зоне находится программируемая плата (см. рис. 8). Изначально к её коннекторам ничего не подключено.

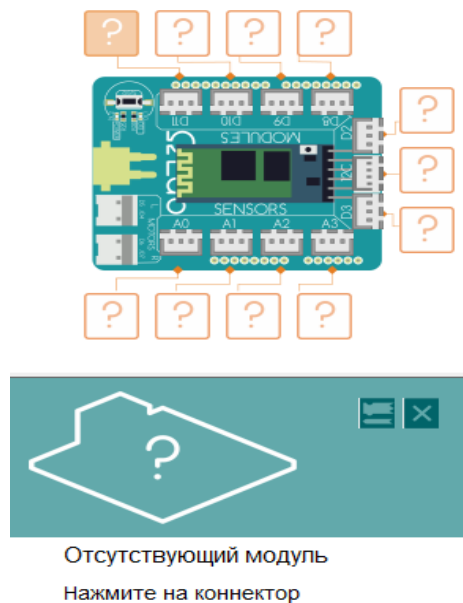
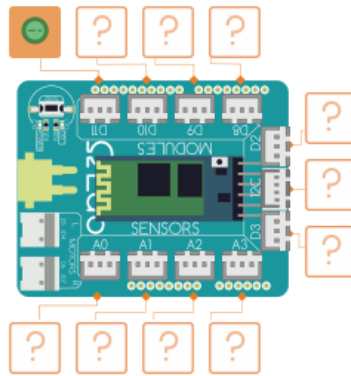


Рисунок 8 – Плата без подключенных устройств/элементов

При нажатии на значок элемента (если он отсутствует, показывается вопросительный знак) в рабочей зоне всплывает описание данного устройства/элемента. Также при нажатии на коннектор для подключения устройства в пятой рабочей зоне всплывает меню с возможными элементами для подключения к плате через данный коннектор. (Рис. 9)

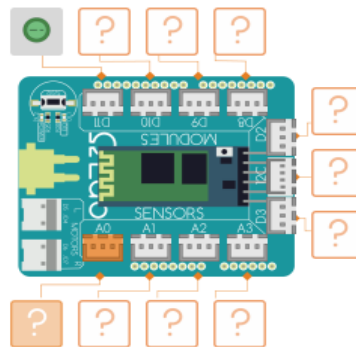


Светодиод

Диод, который светится в диапазоне 0-100, где при 0 значении светодиода не горит, а на 100 горит с максимальной яркостью.

Рисунок 9 – Описание подключенного элемента

На рис. 10 показано меню подключаемых (по возможности) элементов к плате.



- Потенциометр
- Кнопка
- Светодиод
- Геркон
- Магнитный датчик
- Датчик освещенности
- Датчик линии
- Пьезодинамик
- Датчик температуры
- Датчик прикосновения
- Датчик препятствия
- Сервопривод
- Отсутствующий модуль

Рисунок 10 – Меню подключаемых элементов к выбранному коннектору

4. Кнопки для работы с программой.



- Новый. Создает новый пустой проект.



- Сохранить. Пересохраняет текущий проект.



- Сохранить как. Сохраняет проект с возможностью указать имя нового файла.



- Открыть. Открывает ранее сохраненный проект.



- Сгенерировать код. Обратите внимание, что при генерации в всплывающем окне Arduino_IDE появляется код, соответствующий составленной вами программе. Нажмите кнопку «Сгенерировать код». Произойдет компиляция кода. Если блоки подобраны правильно. Это значит, что код будет работать и его можно загрузить на контроллер.



- Загрузить в Arduino. Сохраняет проект, проверяет на возможность компиляции и если компиляция успешна, то проект загружается на плату Arduino.



- Монитор последовательного порта. Открывает Монитор последовательного порта.



- Сохранить как изображение. Сохраняет текущий проект как картинку.



- Открыть сайт. Открывает сайт <https://omegabot.ru>



- Скрыть\открыть Arduino IDE. Скрывает или открывает программу Arduino IDE.

5. Выбор блоков.

Блоки разделяются на три типа: изначальный блок, управляющий блок, блок-команда и блок- переменная.

При этом, при наведении на элемент будет всплывать краткое его описание, например, описание одного из элементов («ЦИФРОВОЙ ПОРТ»): «Считывает числовое значение с контакта». (См. рис. 11).

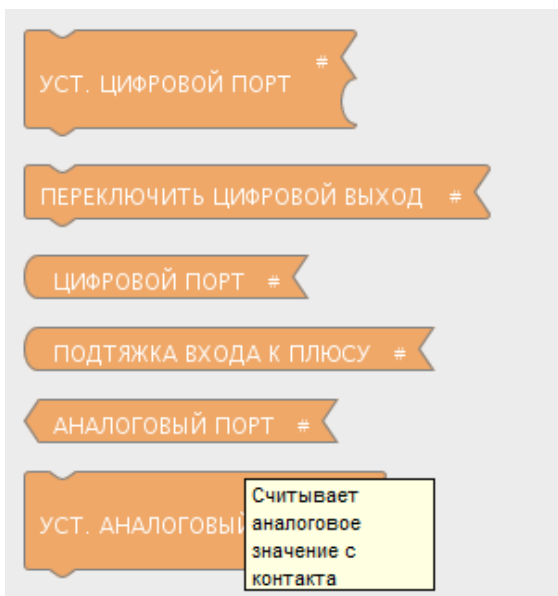


Рисунок 11 – Пример всплывающего описания элемента

6. Для **поиска** нужного блока можно обратиться к «поиску», как на рис. 12. Например, вы можете написать в строку поиска «задержка» и получите блоки-команды, которые выполняют функцию задержки перед следующими командами. Добавьте понравившийся блок в команду.




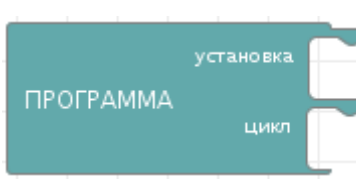
Рисунок 12 – Поисковик элементов (блоков визуального программирования)

Описание модулей программы


Программная среда Omegabot_IDE содержит 16 типов блоков (команд). Типы команд, помимо разного содержания, имеют разный цвет.


1. Управление

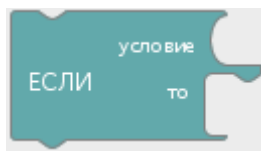
 - Основной цикл, выполняющийся контроллером по умолчанию бесконечно. Все остальные программы должны быть написаны внутри этого цикла.

 - Блок позволяет выполнить ряд настроек в блоке «установка», прежде чем запуститься основной цикл контроллера. В блоке «установка», можно установить первоначальное положение сервопривода или режим светодиода, так же в этом блоке указываются режимы работы для ножек контроллера ввод или вывод.

После загрузки программы на контроллер, сначала один раз выполняются инструкции в блоке «установка», а за тем бесконечно выполняется «цикл».

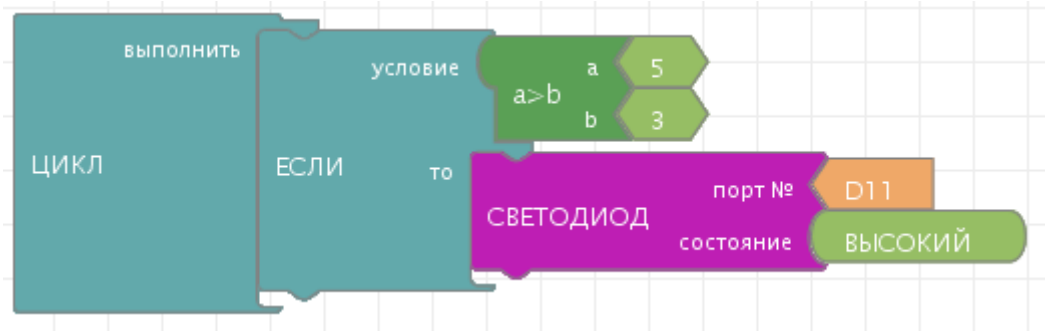
 - Дойдя до этого места, программа остановит свое выполнение на указанное количество миллисекунд. 1 секунда = 1000 миллисекунд.

 - В начале работы контроллера автоматически включается таймер. При помощи данного модуля вы можете получить и использовать текущее значение этого таймера. Модуль возвращает целочисленное значение в миллисекундах. 1 секунда = 1000 миллисекунд.

 - Модуль выполняет логическое выражение. В блоке «условие» вы можете подобрать такие блоки, результатом работы которых будет ответ истина или ложь. В зависимости от этого результата выполняется условие в блоке «то».

Пример:

Если выражение $5 > 3$ истинно, то включить светодиод на порте D11.



ПОКА - Цикл, который начнет выполняться, если выражение в блоке «условие» истинно и продолжит выполняться до тех пор ПОКА «условие» истинно. Выражение в блоке «условие» проверяется после каждой итерации.

ВЫПОЛНЯТЬ ПОКА - Цикл, который сначала выполняет одну итерацию и только потом проверяется ли «условие», если оно истинно, то выполняется еще одна итерация. Модуль будет ВЫПОЛНЯТЬ «команды», до тех пор ПОКА «условие» истинно.

ПОВТОР - Цикл, который будет ПОВТОРЯТЬ блок «команды» строго заданное «количество раз».


ПОВТОРИТЬ И ПОСЧИТАТЬ - Цикл, который выполняя «команды» СЧИТАЕТ «количество раз» ПОВТОРЕНИЙ (итераций) и сохраняет это значение в «переменную». Результат сохраненный в «переменной» можно использовать в других командах модулей.


ПОВТОРЯТЬ МЕЖДУ - Цикл, который выполняя «команды» СЧИТАЕТ «сумму шагов» и сохраняет это значение в «переменную». Цикл будет выполняться в диапазоне от «старт» до «стоп» НЕ включительно. Каждое повторение «переменная» будет увеличиваться на


указанное количество «шагов». Результат сохраненный в «переменной» можно использовать в других командах модулей.

Пример: старт = 0 ; стоп = 10; шагов = 2;

Такой цикл повторится всего 5 раз, каждый раз переменная будет равна 0 – 2 – 4 – 6 и т.д.

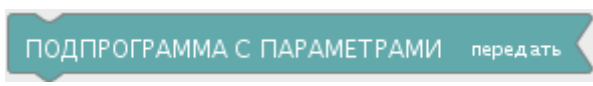
 - Преждевременное прерывание работы любого цикла. Удобно использовать с условием.


 - Создает подпрограмму со своим набором «команд» и модулей. Подпрограммы создаются ВНЕ основного блока программы. Подпрограммы можно вызывать используя условия или как есть в основном блоке программы используя модуль «Подпрограмма» без блока «команды». Двойным щелчком мыши по модулю исправьте наименование вызываемой подпрограммы, в наименовании можно использовать только латинские буквы и цифры, при этом, желательно не начинать его с цифры.

 - Используется для вызова подпрограммы с таким же именем. Двойным щелчком мыши по модулю исправьте наименование вызываемой подпрограммы.

 - Модуль позволяет создать подпрограмму с переменными, которые нужны подпрограмме для ее правильной работы.

Например: ваша подпрограмма вычисляет X. Это значит, подпрограмма должна принимать параметр X, с которым она и будет работать.

 - Вызов подпрограммы с параметрами. В блоке «передать» необходимо вставить те параметры, которые вы указали в вызываемой подпрограмме в качестве принимаемых.

 - Двойным щелчком по модулю перейдите в режим редактирования наименования и укажите здесь имя переменной, передаваемой в качестве параметра в подпрограмму.

2. Порты

УСТ. ЦИФРОВОЙ ПОРТ # - Модуль создает и вызывает подпрограмму с двумя параметрами: 1. Номер цифрового порта
2. Результат логического выражения, которое необходимо передать в цифровой порт на исполнение.

Например:

УСТ. ЦИФРОВОЙ ПОРТ # D11 HIGH
В качестве цифрового порта может быть указан светодиод, а в качестве логического выражения значения high или low, что для светодиода является сигналом включиться или выключиться.

УСТ. АНАЛОГОВЫЙ ПОРТ # - Модуль передает на указанный порт значение для вывода в виде числа.

ПЕРЕКЛЮЧИТЬ ЦИФРОВОЙ ВЫХОД # - Модуль переключает на противоположное значение текущее значение цифрового порта. Например, если светодиод горел, то этот переключатель его выключит, повторное использование этого переключателя включит светодиод.

ЦИФРОВОЙ ПОРТ # - Модуль считывает текущее значение с цифрового порта.

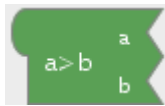
АНАЛОГОВЫЙ ПОРТ # - Читает текущее значение с аналогового порта.

ПОДТЯЖКА ВХОДА К ПЛЮСУ # - Устанавливает переданный пин в режим кнопки и считывает его текущее значение. Режим кнопки – это специальный режим ввода, который будет считывать нажатие и отпускание кнопки.

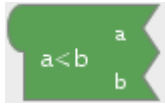
D8 - цифровой порт.

A0 - аналоговый порт.

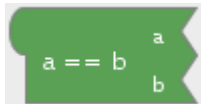
3. Проверки



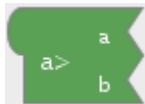
- Выражение вернет true если **a** больше **b**. Модуль принимает числовые значения.



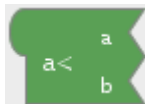
- Выражение вернет true если **a** меньше **b**. Модуль принимает числовые значения.



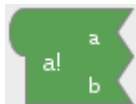
- Выражение вернет true если **a** равно **b**. Модуль принимает числовые значения.



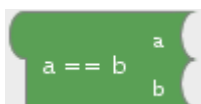
- Выражение вернет true если **a** больше или равно **b**. Модуль принимает числовые значения.



- Выражение вернет true если **a** меньше или равно **b**. Модуль принимает числовые значения.



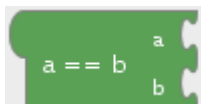
- Выражение вернет true если **a** не равно **b**. Модуль принимает числовые значения.



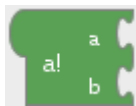
- Выражение вернет true если **a** равно **b** где **a** и **b** это логические выражения.



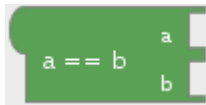
- Выражение вернет true если **a** не равно **b** где **a** и **b** это логические выражения.



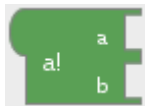
- Выражение вернет true если **a** равно **b** где **a** и **b** это символы. Модуль принимает только символы из таблицы ASCII, так как в виде символа в памяти контроллера может храниться только 1 байт.



- Выражение вернет true если **a** не равно **b**, где a и b это символы. Модуль принимает только символы из таблицы ASCII, так как в виде символа в памяти контроллера может храниться только 1 байт.



- Выражение вернет true если **a** равно **b**, где a и b это строки. Происходит посимвольное сравнение двух строк.



- Выражение вернет true если **a** не равно **b**, где a и b это строки. Происходит посимвольное сравнение двух строк.



- Соединяет два логических выражения логическим оператором «AND».



- Соединяет два логических выражения логическим оператором «OR».



- Инвертирует значение логического выражения. Например: выражение “not(5 == 7)” вернет true.



- Выражение вернет true если a равно b, где a и b это строки. Происходит посимвольное сравнение двух строк.

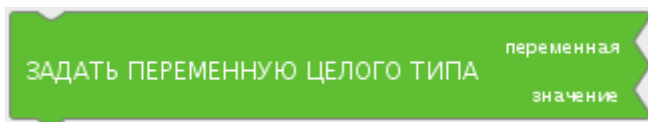


- Проверяет строковую переменную на количество содержащихся в ней символов и возвращает true, если строка пустая.

4. Переменные/Константы



- Целое число



- Модуль позволяет создать и сохранить в памяти **целочисленную переменную**, для дальнейшего её использования в проекте.

Значение переменной может быть число от -32768 до 32767. Попытка записать в переменную число больше или меньше указанного может привести к непредсказуемому результату. Например: Если в переменную записать число 55789 то в результате оно превратиться в число -9747.

ИМЯ ПЕРЕМЕННОЙ ЦЕЛОГО ТИПА

- При помощи модуля можно вызвать по имени ранее заданную **целочисленную переменную**.

ЗАДАТЬ ПЕРЕМЕННУЮ ТИПА ЦЕЛОЕ БЕЗ ЗНАКА

переменная
значение

- Модуль позволяет создать и сохранить в памяти **большую целочисленную переменную без знака минус**, для дальнейшего её использования в проекте. **Значение** переменной может быть число от 0 до 4 294 967 295. Попытка записать в переменную число больше или меньше указанного может привести к непредсказуемому результату.

ИМЯ ПЕРЕМЕННОЙ ТИПА ЦЕЛОЕ БЕЗ ЗНАКА

- При помощи модуля можно вызвать по имени ранее заданную **большую целочисленную переменную без знака минус**.

УСТАНОВИТЬ ЛОГИЧЕСКУЮ ПЕРЕМЕННУЮ

переменная
значение

- Модуль позволяет создать и сохранить в памяти **логическую переменную**, для дальнейшего её использования в проекте. **Значение** переменной может быть ИСТИНА, ЛОЖЬ, ВЫСОКИЙ, НИЗКИЙ или любое другое логическое выражение.

НИЗКИЙ

- Низкий сигнал. Равнозначно логическому ЛОЖЬ (false)

ВЫСОКИЙ

- Высокий сигнал. Равнозначно логическому ИСТИНА (true)

ИСТИНА

- Логическая ИСТИНА.

ЛОЖЬ

- Логическая ЛОЖЬ.

ЗАДАТЬ ВЕЩЕСТВЕННУЮ ПЕРЕМЕННУЮ

переменная
значение

- Модуль позволяет создать и сохранить в памяти **переменную в виде вещественного числа**, для дальнейшего её использования в проекте. **Значение** переменной может быть число от $-3.4028235E+38$ до $3.4028235E+38$ с точностью 6-7 знаков. Попытка записать в переменную число больше или меньше указанного может привести к непредсказуемому результату.

ИМЯ ПЕРЕМЕННОЙ ВЕЩЕСТВЕННОГО ТИПА

- При помощи модуля можно вызвать по имени ранее заданную **переменную** в виде

вещественного числа.

3.1415927

- Вещественное число.

УСТАНОВИТЬ СИМВОЛЬНУЮ ПЕРЕМЕННУЮ

переменная
символ

- Модуль позволяет создать и сохранить в памяти **символьную переменную**, для дальнейшего её использования в проекте. **Значением** переменной могут быть только символы из таблицы ASCII.

ИМЯ СИМВОЛЬНОЙ ПЕРЕМЕННОЙ

- При помощи модуля можно вызвать по имени ранее заданную **символьную переменную**.

A

- Символ.

УСТАНОВИТЬ СТРОКОВУЮ ПЕРЕМЕННУЮ

переменная
значение

- Модуль позволяет создать и сохранить в памяти **строковую переменную**, для дальнейшего её использования в проекте. **Значением** переменной может

быть любой набор символов.

ИМЯ СТРОКОВОЙ ПЕРЕМЕННОЙ

- При помощи модуля можно вызвать по имени ранее заданную **строковую переменную**.

ТЕКСТ СООБЩЕНИЯ

- Строка (набор символов).

СОЗДАТЬ МАССИВ

имя массива
размер

- Модуль позволяет создать и сохранить в памяти **массив целочисленных значений**, для дальнейшего его использования в проекте. Иными словами, это таблица из двух колонок index (порядковый номер строки начинается с нуля) и значение (целое число) где **размер** — это количество таких строк:

index	значение
-------	----------

ИМЯ ПЕРЕМЕННОЙ МАССИВА # - Обратиться к массиву по имени и получить значение в строке #(указать № строки).

ИМЯ МАССИВА
ПОЗИЦИЯ
ЗНАЧЕНИЕ
ЗАДАТЬ ЭЛЕМЕНТ МАССИВА - Обратиться к массиву по **имени** и изменить хранящееся на заданной **позиции** значение на новое указанное **значение**.

5. Математические операторы

$a+b$ a
b - Операция сложения, где a и b целое число.

$a-b$ a
b - Операция вычитания, где a и b целое число.

$a*b$ a
b - Операция умножения, где a и b целое число.

a/b a
b - Операция деления, где a и b целое число.

$a\%b$ a
b - В результате этой операции получится остаток от деления a на b где a и b целое число.

МОДУЛЬ - Операция вычисления модуля из числа. **x**: Если x больше или равен 0.
-x: если x меньше 0.

ОСНОВАНИЕ
ЭКСПОНЕНТА
СТЕПЕНЬ - Операция возведения числа в степень, где **основание** – целое или вещественное число, **экспонента** – степень в которую необходимо возвести **основание**.

КВ. КОРЕНЬ ИЗ - Операция вычисления корня из целого или вещественного числа.

СИНОС (SIN) - Операция вычисления синуса из целого или вещественного числа.

КОСИНУС (COS) - Операция вычисления косинуса из целого или вещественного числа.

ТАНГЕНС (TAN) - Операция вычисления тангенса из целого или вещественного числа.

СЛУЧАЙНЫЙ минимум максимум - Функция возвращает случайное число где **минимум** - нижняя граница случайных значений, включительно, а **максимум** - верхняя граница случайных значений, не включительно.

КАРТА значение от до - Функция пропорционально переносит значения **от** из текущего диапазона значений в новый диапазон **до**. Возможно использование отрицательных значений

В ЦЕЛОЕ значение - функция переводит символьную переменную в целое число.

ГРАНИЦЫ значение нижняя верхняя - Функция проверяет **значение** и если надо задает новое значение, так чтобы оно было в области допустимых значений, заданной параметрами **нижняя граница** и **верхняя граница**. **x**: если **x** входит в область допустимых значений [**a**..**b**]; **a**: если **x** меньше **a**; **b**: если **x** больше **b**.

6. Сенсоры

КНОПКА порт № - Функция возвращает результат работы физического модуля кнопки подключенного к **порту №**. Результатом работы будет всегда значение, т.е. true или false.

ДАТЧИК ОСВЕЩЕННОСТИ порт № - Функция возвращает результат работы физического модуля «датчик освещенности» подключенного к **порту №**. Результатом работы будет целое число от 0 до 1023.

ДАТЧИК ПРЕПЯТСТВИЯ порт № - Функция возвращает результат работы физического модуля «датчик препятствия» подключенного к **порту №**. Результатом работы будет булево значение, т.е. true или false.

ДАТЧИК ТЕМПЕРАТУРЫ порт № - Функция возвращает результат работы физического модуля «датчик температуры» подключенного к **порту №**. Результатом работы будет целое число от 0 до 1023.

ДАТЧИК ХОЛЛА порт № - Функция возвращает результат работы физического модуля «датчик холла» подключенного к **порту №**. Результатом работы будет целое число от 0 до 1023.

СЕНСОРНЫЙ ДАТЧИК порт № - Функция возвращает результат работы физического модуля «сенсорный датчик» подключенного к **порту №**. Результатом работы будет булево значение, т.е. true или false.

ГЕРКОН порт № - Функция возвращает результат работы физического модуля «геркон» подключенного к **порту №**. Результатом работы будет булево значение, т.е. true или false.

ДАТЧИК ЛИНИИ порт № - Функция возвращает результат работы физического модуля «датчик линии» подключенного к **порту №**. Результатом работы будет целое число от 0 до 1023.

ПОТЕНЦИОМЕТР порт № - Функция возвращает результат работы физического модуля «потенциометр» подключенного к **порту №**. Результатом работы будет целое число от 0 до 1023.

УЗК ДАЛЬНОМЕР - Функция возвращает результат работы физического модуля «УЗК Дальномер» подключенного к **порту №**. Результатом работы будет целое число от 0 до 1023.

УЗК ДАЛЬНОМЕР HCSR04 - Функция возвращает результат работы физического модуля «УЗК Дальномер HCSR04» подключенного к **порту №**. Результатом работы будет целое число от 0 до 1023.

ИК ДАТЧИК РАССТОЯНИЯ порт № - Функция возвращает результат работы физического модуля «ИК датчик расстояния» подключенного к **порту №**. Результатом работы будет целое число от 0 до 1023.

ДАТЧИК ЦВЕТА порт № - Функция возвращает результат работы физического модуля «Датчик Цвета» подключенного к **порту №**. В результате получится массив с тремя целочисленными значениями index 0 – Red; index 1 – Green; index 2 - Blue.

7. Серво

СЕРВО порт №, угол - Функция принимает целочисленный параметр **угол** (в градусах). При исполнении, серво привод на **порте №** выполнит наклон\поворот на угол указанный в параметре.

ПЛАВНОЕ СЕРВО порт №
угол
скорость поворота

- Функция принимает целочисленный параметр **угол** (в градусах). При исполнении, серво привод на **порте №** выполнит плавный наклон\поворот на угол указанный в параметре.

8. Светодиоды

СВЕТОДИОД порт №
состояние

- Модуль позволяет управлять светодиодом подключенного к **порту №**. Укажите **состояние** работы светодиода HIGH или LOAD.

МИГАЮЩИЙ ДИОД порт №
количество раз
миллисекунды (горит)
миллисекунды (не горит)

- Модуль позволяет управлять светодиодом подключенного к **порту №**. Укажите **количество раз**, которое светодиод должен мигнуть, а так же задержку в режимах горит\не горит.

СВЕТОДИОД С ЯРКОСТЬЮ порт №
значение

- Модуль позволяет управлять яркостью светодиода подключенного к **порту №**. Укажите **значение** яркости, с которой светодиод должен светить. Принимаются значения от 0 (не горит) до 100(горит с максимальной яркостью).

9. Звук

ЗВУК порт №
тональность звука

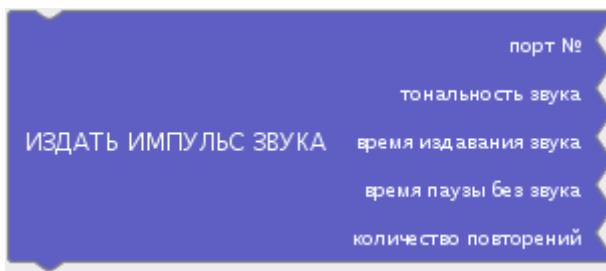
- Позволяет задействовать звуковой модуль на **порту №**. Для выбора **тональности звука** задается в виде числа в Hz.

ЗВУК порт №
тональность звука
время

- Позволяет задействовать звуковой модуль на **порту №**. Для выбора **тональности звука** задается в виде числа в Hz. Звук будет издаваться на протяжении заданного **времени**. Время устанавливается в миллисекундах (1000 мс = 1 с).

ВЫКЛЮЧИТЬ ЗВУК порт №

- Выключает модуль звука на **порту №**.



- Позволяет задействовать звуковой модуль на **порту №**. Для выбора **тональности звука** задается в виде числа в Hz. Звук будет издаваться на протяжении заданного **времени**. Можно так же задать время паузы, и количество повторений. Время устанавливается в миллисекундах (1000 мс =

1 с).

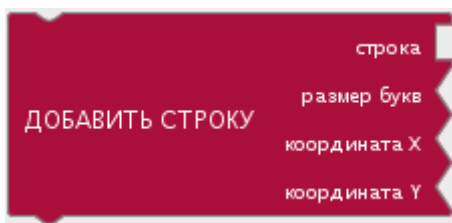
10. Дисплей



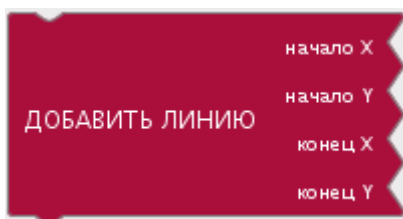
- Очищает дисплей от текущего изображения, функцию рекомендуется вызывать перед отрисовкой нового изображения на дисплее.



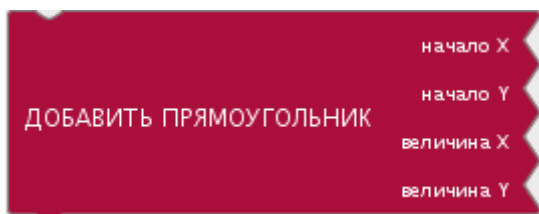
- Выводит подготовленное изображение на дисплей.



- Выводит **строку** на дисплей. Можно задать **размер шрифта**, а так же начальные **координаты** отрисовки текстовой строки.



- Рисует линию на дисплее. В функцию необходимо передать **начальные и конечные координаты**.



- Позволяет изобразить прямоугольник по координатам осей.

11. Ориентация в пространстве

АКСЕЛЕРОМЕТР X - Считывает показания акселерометра по оси X и возвращает их в виде числа.

АКСЕЛЕРОМЕТР Y - Считывает показания акселерометра по оси Y.

АКСЕЛЕРОМЕТР Z - Считывает показания акселерометра по оси Z.

МОДУЛЬ УСКОРЕНИЯ - Считывает показания с датчика и рассчитывает ускорение тела.

ГИРОСКОП X - Считывает и возвращает положение тела на оси X.

ГИРОСКОП Y - Считывает и возвращает положение тела на оси Y.

ГИРОСКОП Z - Считывает и возвращает положение тела на оси Z.

МАГНИТОМЕТР X - Измеряет напряженность магнитного поля по оси X.

МАГНИТОМЕТР Y - Измеряет напряженность магнитного поля по оси Y.

МАГНИТОМЕТР Z - Измеряет напряженность магнитного поля по оси Z.

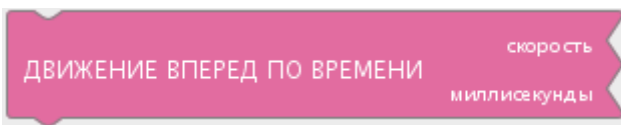
УГОЛ ОТ СЕВЕРА (АЗИМУТ) - Измеряет и возвращает угол поворота тела от Севера.

12. Моторы

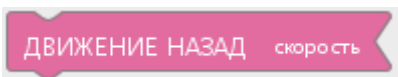
МОТОР левый правый - Модуль позволяет установить скорость вращения для левого и правого мотора.

ДВИЖЕНИЕ ВПЕРЕД скорость - Установите **скорость** вращения моторов.

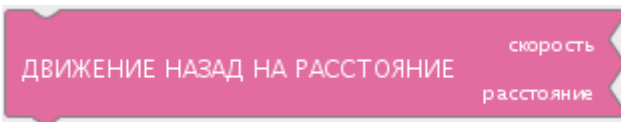
ДВИЖЕНИЕ ВПЕРЕД НА РАССТОЯНИЕ скорость расстояние - Установите **скорость** с которой необходимо двигаться и **расстояние** которое необходимо проехать роботу (только при наличии установленных энкодеров).



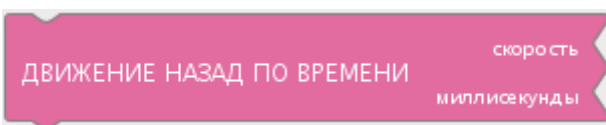
- Установите **скорость** с которой необходимо двигаться и **время в миллисекундах**. Моторы перестанут вращаться спустя указанное время.



- Установите **скорость** вращения моторов.



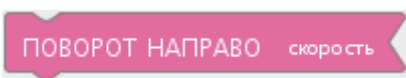
- Установите **скорость** с которой необходимо двигаться и **расстояние** которое необходимо проехать роботу (только при наличии установленных энкодеров).



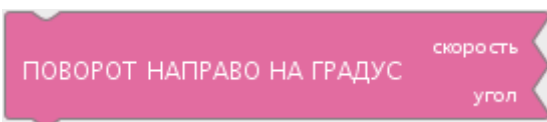
- Установите **скорость** с которой необходимо двигаться и **время в миллисекундах**. Моторы перестанут вращаться спустя указанное время.



- Функция немедленно останавливает моторы.



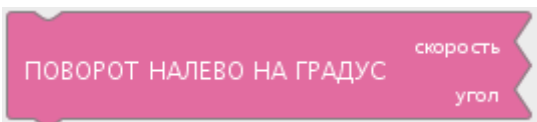
- Робот будет поворачиваться вправо с указанной **скоростью**.



- Совершает поворот на с заданной **скоростью** на установленный **угол**. (только при наличии установленных энкодеров).



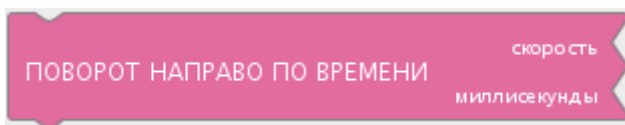
- Робот будет поворачиваться налево с указанной **скоростью**.



- Совершает поворот на с заданной **скоростью** на установленный **угол**. (только при наличии установленных энкодеров).



- Установите **скорость** с которой необходимо двигаться и **время в миллисекундах**. Моторы перестанут вращаться спустя указанное время.



- Установите **скорость** с которой необходимо двигаться и **время в миллисекундах**. Моторы перестанут вращаться спустя указанное время.

13. Эндокеры



- Функция считает каждую 1\40 оборота круга и возвращает количество. Используется энкодер подключенный к ножке D2.



- Функция считывает и возвращает количество пройденных оборотов круга. Используется энкодер подключенный к ножке D2.

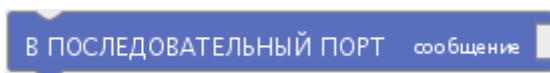


- Функция считает каждую 1\40 оборота круга и возвращает количество. Используется энкодер подключенный к ножке D3.



- Функция считывает и возвращает количество пройденных оборотов круга. Используется энкодер подключенный к ножке D3.

14. Коммуникации



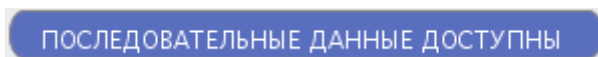
- В качестве **сообщения** функция принимает любое строковое значение и направляет его в **serial port**. Этой функцией удобно пользоваться при отладке. Результат вычисления функций или значения датчиков можно вывести в монитор порта и проверить их правильность.



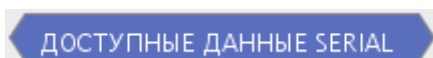
- конвертирует числовое значение в строку.



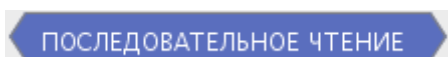
- конвертирует булево значение в строку.



- проверяет буфер serial порта на предмет наличия в нем байтов доступных для чтения. Равносильно Serial.available().



- проверяет буфер serial порта на предмет наличия в нем байтов доступных для чтения. Равносильно Serial.available().



- прочитать один байт из serial порта. После того как байт будет прочитан он удалится из буфера, следующий вызов этой функции прочитает

следующий байт. Таким образом, функцию можно использовать в цикле, чтобы байт за байтом прочитать все данные доступные в буфере.


15. Хранилище

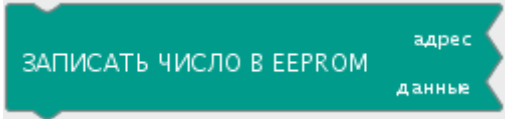
OmegaBot поддерживает работу с **EEPROM**.

EEPROM (англ. Electrically Erasable Programmable Read-Only Memory – электрически стираемое перепрограммируемое постоянное запоминающее устройство (ЭСПЗУ)), она же энергонезависимая память.


Подробнее в официальной документации Ардуино


<https://www.arduino.cc/en/Reference/EEPROM>

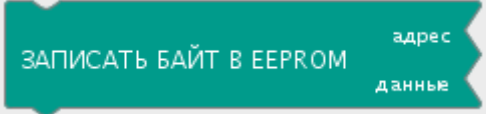
 - Читает число типа int (2 байта) из EEPROM по указанному **адресу**. Из хранилища будет прочитано 2 ячейки от **адрес** до (**адрес** + 1).

 - Сохраняет **данные** типа int(2 байта) в EEPROM хранилище по указанному **адресу**. В хранилище будет задействовано 2 ячейки от **адрес** до (**адрес** + 1).

 - Читает число типа long (4 байта) из EEPROM по указанному **адресу**. Из хранилища будет прочитано 4 ячейки от **адрес** до (**адрес** + 3).

 - Сохраняет **данные** типа long(4 байта) в EEPROM хранилище по указанному **адресу**. В хранилище будет задействовано 4 ячейки от **адрес** до (**адрес** + 3).

 - Читает 1 байт из EEPROM по указанному **адресу**. Из хранилища будет прочитано 1 ячейка по указанному **адресу**.

 - Сохраняет **данные** типа byte в EEPROM хранилище по указанному **адресу**. В хранилище будет задействовано 1 ячейка по указанному **адресу**.

16. Блоки кода

ЗАГОЛОВОК

- Позволяет добавить произвольный код в проект и установить его запуск на момент инициализации глобальных переменных.

```
sketch_sep20a $
```

Ваш код

```
void setup()
{

}

void loop()
{

}
```

УСТАНОВКА

- Позволяет добавить произвольный код в проект и установить его запуск на момент настройки программы.

```
sketch_sep20a $
```

```
void setup()
{
  Ваш код
}

void loop()
{

}
```

ЦИКЛ

- Позволяет добавить произвольный код в проект и установить его запуск на момент работы программы.

```
sketch_sep20a
```

```
void setup()
{

}

void loop()
{
  Ваш код
}
```